C 言語 023 テキストファイル

C 言語でテキストファイルの読み書きを行う手順は下記の通りです。

1. ファイルオープン
2. 読み書きを行う。
3. ファイルクローズ

【ファイルポインタ変数】

```
FILE* 変数名;
```

「FILE 型」はファイルのさまざまな情報が記録されている構造体で「stdio.h」に定義されています。

【ファイルオープンとファイルクローズ】

ファイルオープンには fopen 関数を使用します。

＜fopen 関数＞

定　　義	FILE* fopen(const char* filepath, const char* mode);
ﾍｯﾀﾞｰﾌｧｲﾙ	stdio.h
説　　明	filepath で指定したファイルに、mode で指定したモードでストリームを接続する関数。 モードの指定は「読み込み専用」「書き込み専用」「読み書き両対応」を選択する。
戻 り 値	正常：ファイル情報のアドレス／異常：NULL

ファイルクローズには fclose 関数を使用します。

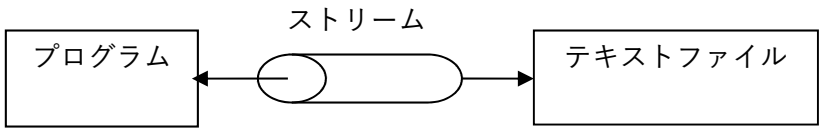
＜fclose 関数＞

定　　義	int fclose(FILE* stream);
ﾍｯﾀﾞｰﾌｧｲﾙ	stdio.h
説　　明	stream のストリームを切断する関数。 ファイルクローズをしなくても、プログラムが終了されれば自動的にクローズされるが、オープン中のファイルは使用に制限がかかるため、使い終わったら明確にファイルクローズを行う事が推奨される。
戻 り 値	正常：0／異常：EOF（EOF はファイルの終端を示す定数）

＜ストリーム＞

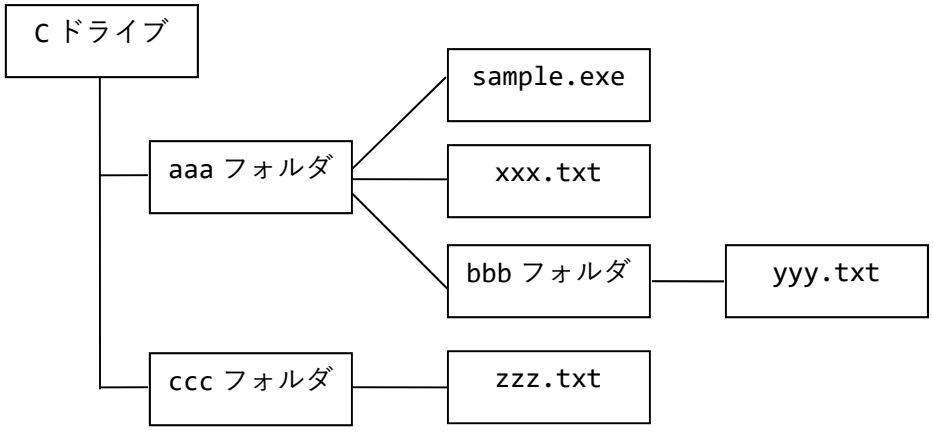
ストリームとは「データの通り道」のような物です。

プログラムとテキストファイルはストリームを介してデータを読み書きします。



＜ファイルのパス＞

ファイルのパスは、直接パスを指定する絶対パスと、exe ファイルがあるフォルダを基点とする相対パスと 2 通りの指定があります。



	xxx.txt	yyy.txt	zzz.txt
絶対パス	c:¥aaa¥xxx.txt	c:¥aaa¥bbb¥yyy.txt	c:¥ccc¥zzz.txt
相対パス	.¥xxx.txt もしくは xxx.txt	.¥bbb¥yyy.txt もしくは bbb¥yyy.txt	..¥ccc¥zzz.txt

文字列にパスを使用する場合、「¥」は「¥¥」にしてエスケープシーケンスすること。

＜モード＞

モード		書き込み方	指定したファイルが存在しなかった場合
r	読み込み専用		関数は異常終了する
w	書き込み専用	上書き	新規作成を行う
a	書き込み専用	追記	新規作成を行う
r+	読み書き両対応	上書き	関数は異常終了する
w+	読み書き両対応	上書き	新規作成を行う
a+	読み書き両対応	追記	新規作成を行う

＜例文＞

```
#include <stdio.h>
int main(void) {
    FILE* fp;                           /* ファイルポインタ変数 */
    fp = fopen("sample.txt", "r");      /* 読み込みモードでファイルオープン */
    if (fp == NULL) {                   /* ファイルオープンエラーの確認 */
        printf("ファイルオープンエラー¥n");
        return -1;                      /* メイン関数を終了する */
    }
    fclose(fp);                         /* ファイルクローズ */
    return 0;
}
```

【ファイルを読み込む】

＜fscanf 関数＞

定　　義	int fscanf(FILE* stream, const char* format, ...);
ﾍｯﾀﾞｰﾌｧｲﾙ	stdio.h
説　　明	ファイルを 1 行（改行コードか EOF まで）読み込み、読み込んだ文字列を scanf 関数の書式指定子を使用して分解し、各変数に代入する関数。
戻 り 値	正常に代入する事ができた値の個数、ファイルの終端を読んだら EOF を返す。

＜例文＞

```
FILE* fp = fopen("sample.txt", "r");
char name[10];
int age;

/* sample.txt を 1 行ずつ読み込んで出力する */
while (fscanf(fp, "%s %d", name, &age) != EOF) {
    printf("%s さんは%d 歳です。¥n", name, age);
}
fclose(fp);
```

「sample.txt」

```
A 太郎 10
B 太郎 20
C 太郎 30
```

＜実行結果＞

```
A 太郎さんは 10 歳です。
B 太郎さんは 20 歳です。
C 太郎さんは 30 歳です。
```

＜fgets 関数＞

定　　義	char* fgets(char *str, int size, FILE* stream);
ﾍｯﾀﾞｰﾌｧｲﾙ	stdio.h
説　　明	ファイルを 1 行（改行コードか EOF まで）か、size-1 バイト分読み込み、読み込んだ文字列を str に代入する関数。
戻 り 値	正常：読み込んだ文字列のアドレス／異常：NULL

＜例文＞

```
FILE* fp = fopen("sample.txt", "r");
char str[11];    /* サイズは 11 バイト */

/* sample.txt を 1 行ずつ読み込んで出力する */
while (fgets(str, 11, fp) != NULL) {    /* サイズは 11 バイト */
    printf("%s", str);                  /* ¥n を入れてない */
}
fclose(fp);
```

「sample.txt」

```
A太郎 10
B太郎 20
C 太郎 30
```

＜実行結果＞

```
A太郎 10
B太郎 20
C 太郎 30
```

printf に¥n を入れてないのに、実行結果が改行されているのは、ファイルから改行コードも読み込んでいるからです。
改行コードは Windows の「CR+LF」の場合 2 バイト。
「A 太郎 10」は A(1)＋太郎(4)＋スペース(1)＋10(2)＋改行コード(2)＝10 バイトになります。
文字配列に記録するには NULL 文字の分も加えて 11 バイト必要になります。

【シーケンシャルとランダム】
ファイルを上から順番に読み込むのを「シーケンシャルアクセス」、任意の場所を読み込むのを「ランダムアクセス」と称します。

【シーケンシャルアクセス】
```
while (fscanf(fp, "%s %d", name, &age) != EOF) {
while (fgets(str, 10, fp) != NULL) {
```
で、ファイルを 1 行ずつ読み込めるのは、ファイルポインタ内に「現在何行目を読んでいるか」を示す「ファイルポジション」を管理しているからです。

→	1 行目
	2 行目
	3 行目
	4 行目
	EOF

fscanf 関数や fgets 関数は実行して読み込む度にファイルポジションをずらす仕様になっています。

【ランダムアクセス】
ランダムアクセスを行うには fseek 関数を使用します。

＜fseek 関数＞

定　　義	int fseek(FILE* stream, long offset, int origin);
ﾍｯﾀﾞｰﾌｧｲﾙ	stdio.h
説　　明	stream のファイルポジションを、ファイル内の origin（起点）から offse バイト数分ずらした場所に移動させる関数。
戻　り　値	正常：0／異常：0 以外

＜起点で使用する定数＞

SEEK_SET	ファイルの先頭
SEEK_CUR	ファイルの現在位置
SEEK_END	ファイルの終端

＜例文＞

sample.txt のデータが左記であれば、実際のデータは右記になる。

sample.txt	実際のデータ
aaaaa bbbbb ccccc ddddd	aaaaa[¥n]bbbbb[¥n]ccccc[¥n]ddddd[EOF]

（[¥n]は改行コード、EOF はファイルの終了を示す）

半角文字は 1 バイト、改行コードは Windows の「CR+LF」の場合 2 バイト。
3 行目を読み込みたい場合は、ファイルの先頭から 14 バイト目にファイルポジションを合わせます。(先頭は 0 バイト目）

```
FILE* fp = fopen("sample.txt", "r");
char str[8];                /* 5 文字＋改行コード＋NULL 文字の 8 バイト */
fseek(fp, 14, SEEK_SET);    /* ファイルポジションをファイル先頭から 14 バイト目にする */
fgets(str, 8, fp);          /* fgets 関数、fscanf 関数で 14 バイト目から改行コードまで読む */
printf("%s", str);          /* 「ccccc」と出力 */
fclose(fp);
```

現在のファイルポジションの位置を知りたい場合は ftell 関数を使用します。

＜ftell 関数＞

定　　義	long ftell(FILE* stream);
ﾍｯﾀﾞｰﾌｧｲﾙ	stdio.h
説　　明	stream のファイルポジションの位置を取得する関数。
戻　り　値	正常：ファイルポジションの位置／異常：-1

【ファイルに書き込む】

＜fprintf 関数＞

定　　義	int fprintf(FILE* stream, const char* format, ...);
ﾍｯﾀﾞｰﾌｧｲﾙ	stdio.h
説　　明	printf 関数の書式指定子を使用して stream に文字列を書き込む関数。
戻 り 値	正常：書き込んだ文字列のバイト数／異常：負数

＜例文＞

```
FILE* fp = fopen("sample.txt", "w");    /* 上書きモード */
char name[] = "A 太郎";
int age = 10;
fprintf(fp,"%s さんは%d 歳です。¥n", name, age);
fclose(fp);
```

「sample.txt」
A 太郎さんは 10 歳です。

＜fputs 関数＞

定　　義	int fputs(const char* str, FILE *stream);
ﾍｯﾀﾞｰﾌｧｲﾙ	stdio.h
説　　明	stream に str を書き込む関数。
戻 り 値	正常：正数／異常：EOF

＜例文＞

```
FILE* fp = fopen("sample.txt", "a");    /* 追記モード */
char str[100];
scanf("%s", str);    /* コマンドプロンプトから入力した文字列を */
fputs(str, fp);      /* ファイルに書き込む */
fclose(fp);
```

＜モードによる書き込み方＞

書き込み方には「上書き」と「追記」があり、ファイルオープン時に指定します。
「上書き」はファイルのデータを全部消去した後に、データの書き込みを行います。
「追記」はファイルの末尾にデータの書き込みを行います。
また、指定したファイルが無かった場合、エラーになるか、ファイルを新規作成するかモードによって異なるので、用途に合わせて選択してください。

【CSV ファイルの読み書き】

```
#include <stdio.h>
#include <string.h>    /* strtok 関数を使用するため */
#include <stdlib.h>    /* atoi 関数を使用するため */

int main(void) {

    /* 変数宣言 */
    FILE* fp;          /* ファイルポインタ */
    char buff[20];    /* バッファ用変数 */
    char name[10];    /* 名前 */
    int age;           /* 年齢 */

    /* 入力用データ */
    char name1[] = "A 太郎";
    int age1 = 10;
    char name2[] = "B 太郎";
    int age2 = 20;
    char name3[] = "C 太郎";
    int age3 = 30;

    /* CSV の書き込み */
    fp = fopen("sample.csv", "w");
    fprintf(fp,"%s,%d¥n", name1, age1);
    fprintf(fp,"%s,%d¥n", name2, age2);
    fprintf(fp,"%s,%d¥n", name3, age3);
    fclose(fp);

    /* CSV の読み込み */
    fp = fopen("sample.csv", "r");
    while (fgets(buff, 1024, fp) != NULL) {

        /* strtok で文字列を分解 */
        strcpy(name,strtok(buff, ","));
        age = atoi(strtok(NULL, ","));
```

（続く）

```
        printf("名前：%s 年齢：%d¥n", name, age);

    }
    fclose(fp);

    return 0;
}
```